In-term exam 3 NEXT TUESDAY

CSE 8B Today

Listeners and Events in a very simple GUI

PSA 6!
- Released tonight
- Due next Wednesday!!
- TODAY and THURSDAY: HELP to work on it

Seating requests:
IF YOU NEED a NEW REQUEST SUBMIT BY THURSDAY night!! WE ARE CHANGING NEXT WEEK
Exam 3: Next Tuesday(!!)

• Review at Tomorrow’s Discussion. This test will be more questions and shorter

• Focus topics/skills
  – Write a class that extends another class
  – Write a method that overrides a method in a base class
  – Reason about what will happen when calling methods in subclass/super-class objects
  – Describe how Java uses the type of the object at run-time to decide which method to call (Polymorphism)
  – Know when a reference to an object is legal, and when it will cause a compile error (e.g. a Person reference pointing to a Student object vs. a Student reference pointing to a Person object)
  – Identify and describe situations where casting is needed (and safe, or not safe). Examples where runtime errors happen?
  – Use the Graphics object to draw simple scenes
  – Describe how the paintComponent method is used to provide custom graphics in GUIs
  – **Use Interfaces and Abstract classes**, and describe what these are and when you would use them
  – Reason about errors when using interfaces and abstract classes

• NOT included:
  – Events
  – Inner classes
PSA6: Connect 4 GUI

PSA 6: a NICE INTERFACE FOR A GAME!
- your ConnectFourBoard class controls game play and game state (AND IT's ALREADY DONE!)
- Your *interface simply calls methods* *(NO NEED TO RE-IMPLEMENT the connectFourBoard class!)*
*You CAN just USE the methods we give* on the board to make moves and reflect changes in the game.

Two new central concepts:
1. **Today**: Writing and registering Listener objects (Event-driven programming)
2. **Thursday**: Inner classes
   (Not to be confused with subclasses!!)

Today and Thursday: working on the code you will work with and write for your PSA. **However**, today we won’t talk much about inner classes. You can still get started on PSA6, but make sure you do the reading for Thursday first.
public class ConnectFourSimpleDemo extends JFrame {
    
    public ConnectFourSimpleDemo() {
        this.turn = 'X';
        JButton jbtReset = new JButton( "New Game" );
        JButton jbtSwitch = new JButton( "Switch turns" );

        JPanel buttonPanel = new JPanel();
        buttonPanel.add( jbtReset );
        buttonPanel.add( jbtSwitch );

        this.status = new JLabel( "Welcome to Connect 4! Turn is: " + this.turn );

        JPanel displayBoard = new JPanel();
        JPanel p1 = new MyPanel();
        JPanel p2 = new MyPanel();
        JPanel p3 = new MyPanel();

        displayBoard.setLayout( new GridLayout( 1, 3 ) );
        displayBoard.add( p1 );
        displayBoard.add( p2 );
        displayBoard.add( p3 );

        this.setLayout( new BorderLayout() );
        this.add( this.status, BorderLayout.NORTH );
        add( displayBoard, BorderLayout.CENTER );
        add( buttonPanel, BorderLayout.SOUTH );

        pack();
        setVisible( true );
    }
}

You should be able to sketch what this GUI will look like
public ConnectFourSimpleDemo()
{
    this.turn = 'X';
    JButton jbtReset = new JButton( "New Game" );
    JButton jbtSwitch = new JButton( "Switch turns" );

    JPanel buttonPanel = new JPanel();
    buttonPanel.add( jbtReset );
    buttonPanel.add( jbtSwitch );

    this.status = new JLabel( "Welcome to Connect 4!  Turn is: " + this.turn );

    JPanel displayBoard = new JPanel();
    JPanel p1 = new MyPanel();
    JPanel p2 = new MyPanel();
    JPanel p3 = new MyPanel();

    displayBoard.setLayout( new GridLayout( 1, 3 ) );
    displayBoard.add( p1 );
    displayBoard.add( p2 );
    displayBoard.add( p3 );

    this.setLayout( new BorderLayout() );

    add( this.status, BorderLayout.NORTH );
    add( displayBoard, BorderLayout.CENTER );
    add( buttonPanel, BorderLayout.SOUTH );

    pack();
    setVisible( true );
}

class MyPanel extends JPanel
{
    protected void paintComponent( Graphics g )
    {
        g.setColor( Color.yellow );
        g.fillOval(0, 0, getWidth(), getHeight());
    }

    public Dimension getPreferredSize()
    {
        return new Dimension( 100, 200 );
    }
}
class ButtonListener implements ActionListener
{
    public void actionPerformed( ActionEvent e )
    {
        System.out.println( "Button clicked!" );
    }
}

When a button is clicked, Java generates an ActionEvent on the button. It automatically calls the actionPerformed method on the listener(s) registered with that button.
Adding a Listener

```java
public ConnectFourSimpleDemo()
{
    this.turn = 'X';
    JButton jbtReset = new JButton( "New Game" );
    JButton jbtSwitch = new JButton( "Switch turns" );
    ButtonListener resetListener = new ButtonListener();

    JPanel buttonPanel = new JPanel();
    buttonPanel.add( jbtReset );
    buttonPanel.add( jbtSwitch );

    ...}
```

1st step: CREATE CODE TO BE RUN WHEN THE ACTION HAPPENS! (previous slide)

2nd step: CREATE ONE INSTANCE of our “LISTENER” (otherwise there is no method to be run)

If we add the line in red to the code, what will the GUI do when we click on the buttons?

A. It will print “Button clicked!” when either the New Game or the Switch turns button is clicked.
B. It will print “Button clicked!” when the New Game button is clicked, but not when the Switch turn button is clicked.
C. Nothing will happen when either button is clicked.
D. It will have an error.
public ConnectFourSimpleDemo()
{
  ...
  JPanel displayBoard = new JPanel();
  JPanel p1 = new MyPanel();
  p1.addMouseListener( new PanelClickListener() );
  ...
}

class PanelClickListener implements MouseListener
{
  public void mouseClicked( MouseEvent e )
  {
    System.out.println( "Clicked on a label" );
  }
}
  // end of the MouseListener class

What will our GUI do if we add the code above?
A. Nothing it didn’t already do
B. It will print “Clicked on a label” when we click on the leftmost yellow oval.
C. It will print “Clicked on a label” when we click on any of the yellow ovals.
D. Nothing, there is an error with this code.
Making our GUI more interesting

Each oval represents a “space” in a 3-position “board”
Yellow indicates that no one has played.

If it is ‘X’’s turn, clicks should turn the ovals blue

If it is ‘O’’s turn, clicks should turn the ovals green

Turns will switch only when the user clicks “Switch turns”

Idea: We will create a new instance variable “board” and the Panels will paint based on the contents of the “board” variable
public class ConnectFourSimpleDemo extends JFrame {
    private char turn;
    private char[] board;
    private JLabel status;

    public ConnectFourSimpleDemo() {
        this.turn = 'X';  // Initialize turn to X
        this.board = new char[3];  // Initialize the board (empty)
        this.board[0] = ' ';
        this.board[1] = ' ';
        this.board[2] = ' ';

        JPanel buttonPanel = makeButtonPanel();
        JPanel displayBoard = makeDisplayBoard();
        this.status = new JLabel( "Welcome to Connect 4! Turn is: " + this.turn);

        setLayout( new BorderLayout() );
        add( this.status, BorderLayout.NORTH );
        add( displayBoard, BorderLayout.CENTER );
        add( buttonPanel, BorderLayout.SOUTH );

        pack();
        setVisible( true );
    }
}
public class ConnectFourSimpleDemo extends JFrame {
// ConnectFourSimpleDemo is a completely separate class, even though
// both classes are defined in the same file.
} END of ConnectFourSimpleDemo

class MyPanel extends JPanel {
private int position;

public MyPanel( int pos ){
    position = pos;
}

protected void paintComponent( Graphics g ){
    super.paintComponent(g);
    if ( board[position] == 'X' )
        g.setColor( Color.blue );
    else if ( board[position] == 'O' )
        g.setColor( Color.green );
    else
        g.setColor( Color.yellow );
    g.fillOval( 0, 0, getWidth(), getHeight() );
}

Will this work?
A. Yes
B. No, because you cannot access the
   ConnectFourSimpleDemo
   member variable board from the
   MyPanel class
C. No, because the MyPanel
   constructor does not explicitly call the
   constructor of the superclass (JPanel)
I could pass a LOT of parameters to the constructor...
instead of just `public MyPanel( int pos )` BUT . . . THERE IS A BETTER SOLUTION
Inner Classes

ConnectFourSimpleDemo object

- turn: ‘X’
- status: Address of the JLabel object
- board

MyPanel object

- position

Now if Java can’t find a variable in the MyPanel object, it will look for it in the CFSD object!

Do not confuse this with subclasses!
The MyPanel object exists within the scope of the CFSD object, but there is no subclass relation. The board variable is still in the CFSD object, NOT the MyPanel object.
ConnectFourSimpleDemo object

- turn: ‘X’
- status: Address of the JLabel object
- board:

MyPanel objects

- position 0
- position 1
- position 2

Now if Java can’t find a variable in the MyPanel object, it will look for it in the CFSD object!

Do not confuse this with subclasses! The MyPanel object exists within the scope of the CFSD object, but there is no subclass relation. The board variable is still in the CFSD object, NOT the MyPanel object. A MyPanel exists within a CFSD, but it is NOT a CFSD.
public class ConnectFourInnerDemo extends JFrame {
   // ConnectFourInnerDemo class defined here . . . 
   class MyPanel extends JPanel //inner from CFID
   {
      private int position; // A new variable in MyPanel
      protected void paintComponent(Graphics g) {
         super.paintComponent(g);
         if (board[position] == 'X')
            g.setColor(Color.blue);
         else if ( board[position] == 'O' )
            g.setColor(Color.green);
         else g.setColor(Color.yellow);
         g.fillOval(0, 0, getWidth(), getHeight());
      }
      // other MyPanel methods defined here
   }
   // Other methods defined here
   
   class PanelClickListener implements MouseListener
   {//inner from MyPanel
      public void mouseClicked(MouseEvent e)
      {
         ________________________________;
         repaint();
      }
      // Other methods defined here
   }
   How many “inner” levels??
ConnectFourInnerDemo object

- turn: ‘X’
- status: Address of the JLabel object
- board

MyPanel object

- position: 0
- PanelClickListener object

PanelClickListener object

- public void mouseClicked(…)
  // writes data to board

// method from MyPanel object
protected void paintComponent(Graphics g)
{
  super.paintComponent(g);
  if (board[position] == 'X')
    g.setColor(Color.blue);
  else if (board[position] == 'O')
    g.setColor(Color.green);
  else
    g.setColor(Color.yellow);
  g.fillOval(0, 0, getWidth(), getHeight());
}

// reads data from board
public class ConnectFourInnerDemo extends JFrame {
    // CFID defined here
    class MyPanel extends JPanel {
        private int position; // A new variable in MyPanel
        // MyPanel methods defined here

        class PanelClickListener implements MouseListener {
            public void mouseClicked( MouseEvent e ) {
                ____________________________;
                repaint();
            }
            // We will register this listener
            // with the MyPanel objects
        }
    }
}

What should go in the blank to make the panel turn blue when the user clicks on it and it’s X’s turn?
A. board[position] = turn;
B. g.setColor( Color.blue );
C. if ( turn == ‘X’ ) g.setColor( Color.blue );
D. Something else

HINT: calls to paintComponent
protected void paintComponent( Graphics g )
{
    super.paintComponent(g);
    if (board[position] == 'X' )
        g.setColor( Color.blue );
    else if ( board[position] == 'O' )
        g.setColor( Color.green );
    else
        g.setColor( Color.yellow );
    g.fillOval(0, 0, getWidth(), getHeight());
}

public void mouseClicked( MouseEvent e )
{
    board[position] = turn;
    repaint();
}
Why do we need to call `repaint` in the `mouseClicked` method? (discuss)
Adding the PanelClickListener

ConnectFourInnerDemo object (extends JFrame)

turn

‘X’

status

Address of the JLabel object

board

Address of the board array

MyPanel object (extends JPanel)

position

0

PanelClickListener object (implements MouseListener)

public void mouseClicked( ... )

public ConnectFourInnerDemo()
{
    ...
    JPanel displayBoard = new JPanel();
    JPanel p1 = new MyPanel();
    p1.addMouseListener(new PanelClickListener());
    ...
}

Why won’t this code work?
A. The PanelClickListener must be instantiated inside a MyPanel object
B. A PanelClickListener is not a MouseListener
C. PanelClickListener objects cannot be instantiated using new because PanelClickListener is an interface